DYNAMICALLY EXCHANGING COMPUTER USER'S CONTEXT

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of provisional U.S. Patent Application No. 60/194,006 ~~(Attorney Docket No. 294438002US)~~, entitled "DYNAMICALLY EXCHANGING COMPUTER USER'S CONTEXT" and filed April 2, 2000, and of provisional U.S. Patent Application No. 60/193,999 ~~(Attorney Docket No. 294438008US)~~, entitled "OBTAINING AND USING CONTEXTUAL DATA FOR SELECTED TASKS OR SCENARIOS, SUCH AS FOR A WEARABLE PERSONAL COMPUTER" and filed April 2, 2000. These applications are both hereby incorporated by reference in their entirety.

TECHNICAL FIELD

The following disclosure relates generally to computer-based modeling of information, and more particularly to modeling and exchanging context data, such as for a wearable personal computer.

BACKGROUND

Computer systems increasingly have access to a profusion of input information. For example, a computer may be able to receive instructions and other input from a user via a variety of input devices, such as a keyboard, various pointing devices, or an audio microphone. A computer may also be able to receive information about its surroundings using a variety of sensors, such as temperature sensors. In addition, computers can also receive information and communicate with other devices using various types of network connections and communication schemes (*e.g.*, wire-based, infrared or radio communication).

wearable, the surrounding physical environment and/or the available electronic data environment (or "cyber-environment"). In some embodiments the context is represented (or "modeled") with a variety of attributes (or "variables") each modeling a single aspect of the context. By facilitating the exchange of context information, the facility reduces dependencies of context client applications on specific sensors and other input sources, resolves conflicts between context client applications that consume the same context data, resolves conflicts between multiple sensors or other input sources that produce the same data, and isolates the generation of derived attributes from context client applications.

A context is modeled or represented with multiple attributes that each correspond to a specific element of the context (*e.g.*, ambient temperature, location or a current user activity), and the value of an attribute represents a specific measure of that element. Thus, for example, for an attribute that represents the temperature of the surrounding air, an 80° Fahrenheit value represents a specific measurement of that temperature. Each attribute preferably has the following properties: a name, a value, an uncertainty level, units, and a timestamp. Thus, for example, the name of the air temperature attribute may be "ambient-temperature," its units may be degrees Fahrenheit, and its value at a particular time may by 80. Associated with the current value may be a timestamp of 02/27/99 13:07 PST that indicates when the value was generated, and an uncertainty level of +/- 1 degrees.

Context servers supply values for attributes by receiving and processing input information from sensors or other sources. Attribute values provided by a context server may either be "measured" (or "observed") in that they are directly received from an input source, or may instead be "derived" in that they are the result of performing processing on one or more measured attribute values. Indeed, a derived attribute value may be produced by performing additional processing on one or more other derived attribute values. Context attributes (or "condition variables") are discussed in greater detail in both U.S. Patent Application No. 09/216,193, filed December 18, 1998 and entitled "METHOD AND SYSTEM FOR CONTROLLING PRESENTATION OF INFORMATION TO A USER BASED ON THE USER'S CONDITION", Pat. No. 6,466,232, and provisional U.S. Patent Application No. 60/193,999 ~~(Attorney Docket No.~~

~~29443800SUS~~, filed April 2, 2000 and entitled "OBTAINING AND USING CONTEXTUAL DATA FOR SELECTED TASKS OR SCENARIOS, SUCH AS FOR A WEARABLE PERSONAL COMPUTER," which are both hereby incorporated by reference.

When the characterization module obtains an attribute value from a context server, it caches the value for use when responding to future requests from context clients for a value of the attribute. Thus, when the characterization module receives a request from a context client for the value of an attribute, the characterization module determines whether it has a cached value for the attribute and, if so, whether the value is sufficiently accurate (*e.g.*, the value does not have too high of an uncertainty) and/or sufficiently recent (*e.g.*, the value is not too old). If the value is not sufficiently accurate or recent, the characterization module requests and receives an updated value for the attribute from the context server that supplied the value. When the characterization module has a sufficiently accurate and recent value, it supplies the value to the context client. The determination of whether a value is sufficiently accurate and recent can be made in a variety of ways, such as by using thresholds for recency or uncertainty that can be specified by the context client during the request, by a context server for all values of an attribute or for a specific attribute value, or by the characterization module.

In some embodiments, two or more different context servers may supply to the characterization module their own distinct values for a single attribute. For example, a first context server can supply a value for a user.location attribute based on data received from a global positioning system device, while a second context server can supply a value for the user.location attribute based on data received from an indoor positioning device. Alternately, the first and second context servers could use the same input information when determining the value for a single attribute, but could use different methods to perform the determining and could thus arrive at different values. When multiple content servers supply values for the same attribute, each of the context servers is said to supply values for a separate "instance" of the attribute. The characterization module preferably provides a variety of different approaches, called

that consumes a particular attribute value may also create a condition in the characterization module (not to be confused with the current modeled condition of the user or the environment that is represented by various attribute values) for testing that attribute by calling a CreateCondition function. Once a context client has created a condition, it can evaluate the condition by calling an EvaluateCondition function using parameters to identify the condition, and may also proceed to create a condition monitor that monitors the condition and notifies the context server when the condition is satisfied by calling a CreateConditionMonitor function. To suspend operation of a created condition monitor, a context server may call a StopConditionMonitor function, and to resume its operation, may call a StartConditionMonitor function. The context server may remove a condition monitor that it created by calling a RemoveConditionMonitor function and, correspondingly, may remove a condition that it created by calling a RemoveCondition function. A context client may unregister with the characterization module by calling an UnregisterContextClient function. A context server may similarly remove attribute instances that it has registered by calling a RemoveAttributeInstance function. Before it does, however, it may first call a CheckAttributeInstanceDependencies function to determine whether any context clients currently depend upon that attribute instance. A context server may unregister with the characterization module by calling an UnregisterContextServer function. A set of API functions are discussed in greater detail in both U.S. Patent Application No. 09/541,328, filed April 2, 2000 and entitled "INTERFACE FOR EXCHANGING CONTEXT DATA," abandoned, and provisional U.S. Patent Application No. 60/194,123, filed April 2, 2000 and entitled "SUPPLYING AND CONSUMING USER CONTEXT DATA," which are both hereby incorporated by reference.

In some embodiments, it may also be useful to store attribute value information in a more permanent fashion than a temporary cache. For example, it may be useful for the characterization module to keep a log of all attribute values received and sent, or of all interactions with context clients and context servers. Alternately, it may be useful to record the current values of some or all of the attributes and attribute instances at the same time, such as to capture a complete model of the current context. Storing

attribute value information is discussed in greater detail in both U.S. Patent Application No. 09/464,659, filed December 15, 1999 and entitled "STORING AND RECALLING INFORMATION TO AUGMENT HUMAN MEMORIES", Pat. No. 6,513,046, and U.S. Patent Application No. 09/541,326, filed April 2, 2000 and entitled "LOGGING AND ANALYZING COMPUTER USER'S DATA," abandoned, which are both hereby incorporated by reference. Other uses of attribute value information are described in provisional U.S. Patent Application No. 60/194,000, filed April 2, 2000 and entitled "SOLICITING PRODUCT INFORMATION BASED ON THE USER'S CONTEXT," in provisional U.S. Patent Application No. 60/194,002, filed April 2, 2000 and entitled "AUTOMATED SELECTION OF UNSOLICITED INFORMATION BASED ON A USER'S CONTEXT," and in provisional U.S. Patent Application No. 60/194,758, filed April 2, 2000 and entitled "CREATING PORTALS BASED ON THE USER'S CONTEXT," each of which are hereby incorporated by reference.

Figure 1 illustrates an embodiment of the characterization module which executes on a general-purpose body-mounted wearable computer 120 worn by user 110. Many wearable computers are designed to act as constant companions and intelligent assistants to a user, and are often strapped to a user's body or mounted in a holster. The computer system may also be incorporated in the user's clothing, be implanted in the user, follow the user, or otherwise remain in the user's presence. In one preferred embodiment the user is human, but in additional preferred embodiments, the user may be an animal, a robot, a car, a bus, or another entity whose context is to be modeled. Indeed, the computer system may have no identifiable user, but rather operate as an independent probe, modeling and/or reporting on the context in an arbitrary location.

The wearable computer 120 has a variety of user-worn user input devices including a microphone 124, a hand-held flat panel display 130 with character recognition capabilities, and various other user input devices 122. Similarly, the computer has a variety of user-worn output devices that include the hand-held flat panel display, an earpiece speaker 132, an eyeglass-mounted display 134, and a tactile display 136. In addition to the various user-worn user input devices, the computer can also receive information from various user sensor input devices 116 and from environment